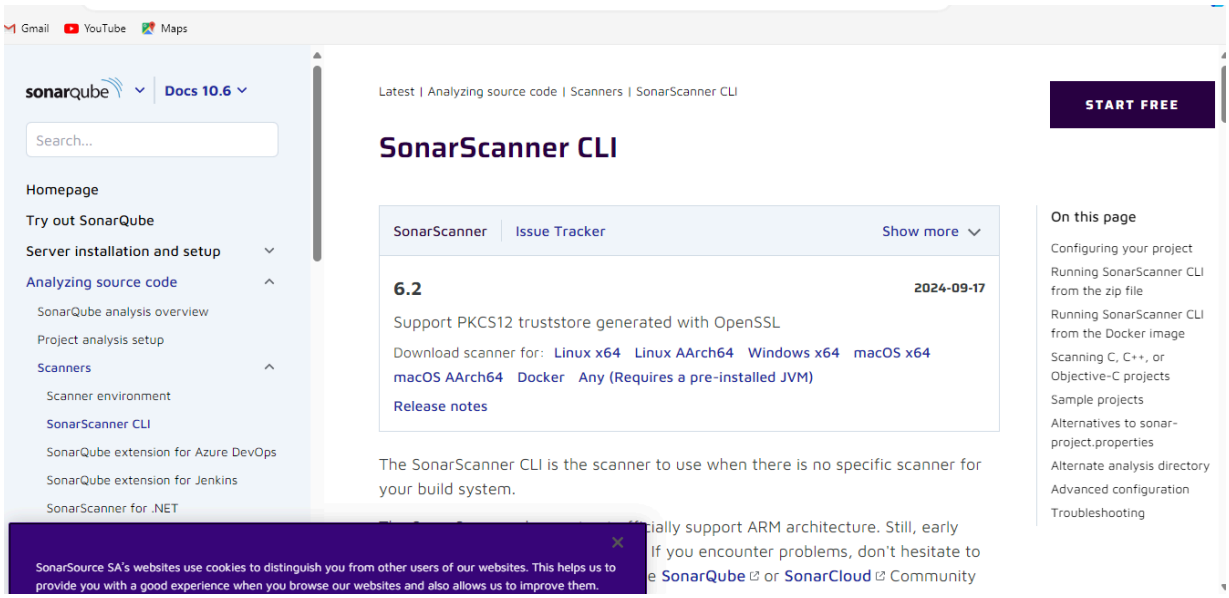
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 08

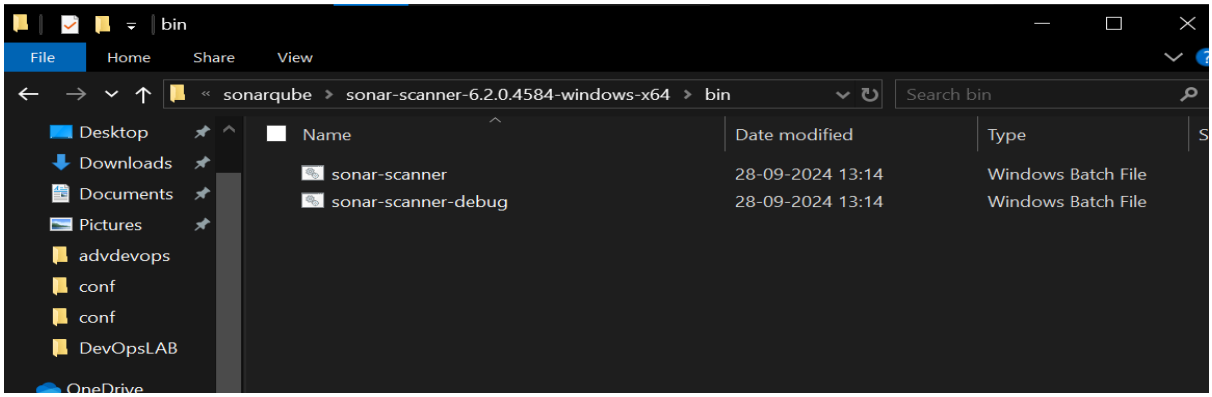
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web /Java / Python application.
(I have performed this experiment entirely on my laptop)

For the Docker installation problems that i was facing during experiments 6 and 7, I resorted to the option of Resetting Windows which helped me clear the corrupted files which were causing my Laptop's Restart issues

Step 1: Download sonar scanner. We will require it in the further steps while building our pipeline
SonarScanner CLI (sonarsource.com)
Visit this link and download the sonarqube scanner CLI.

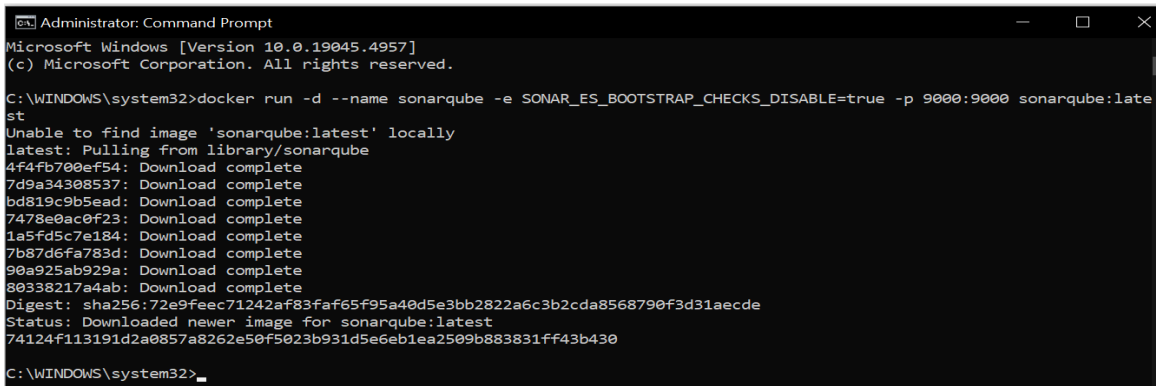


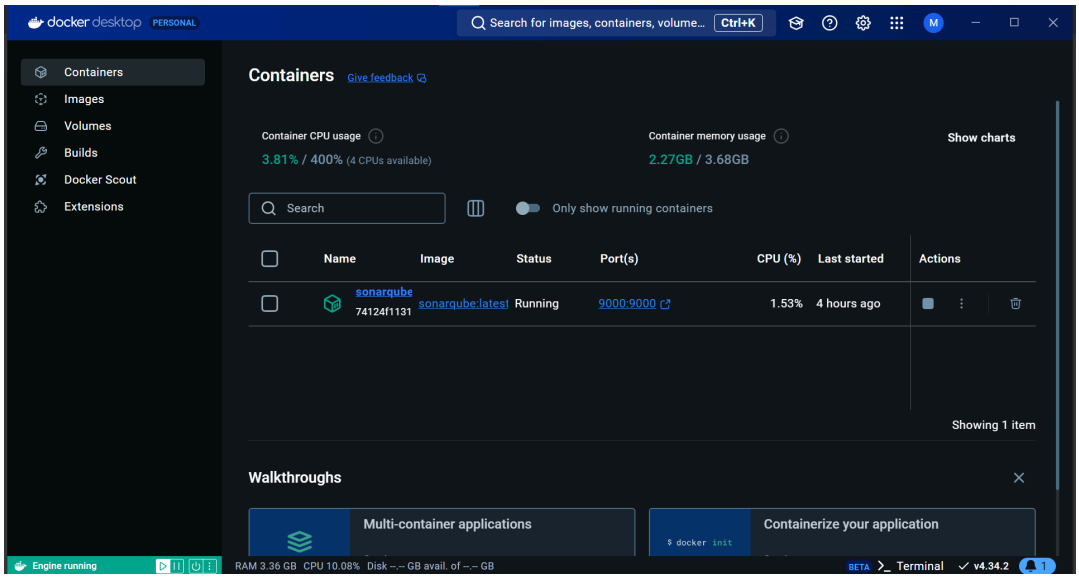
Extract the zip files in a folder



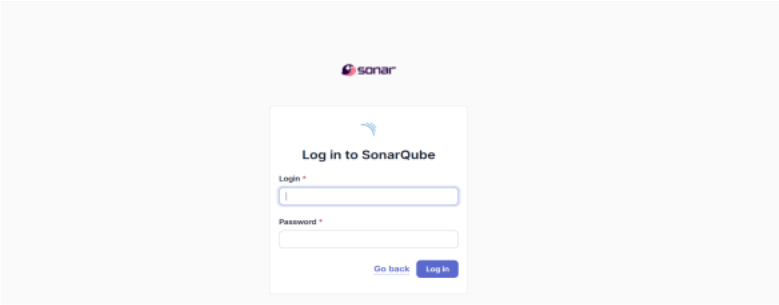
Name: Mohit Kerkar Div: D15C Roll No: 23

Run this command in an administrative command prompt so as to create a sonarqube image on localhost:9000



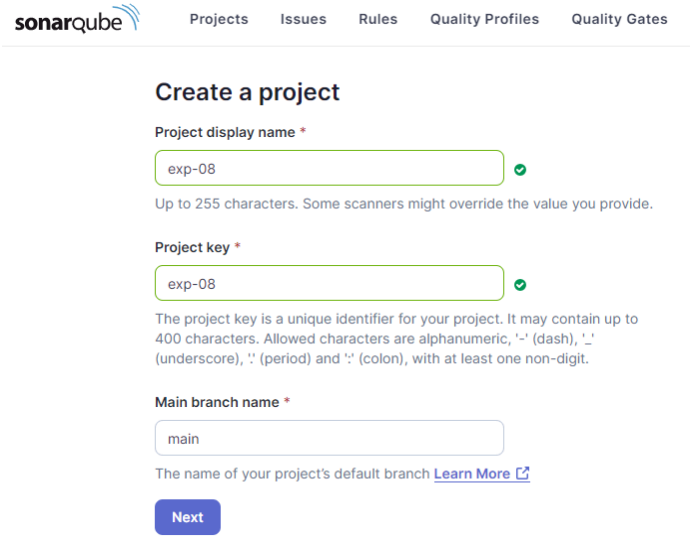


Visit localhost:9000 on your browser and login to your account(the default credentials are admin and admin for username and password. It allows to change it and set our own password for our account)

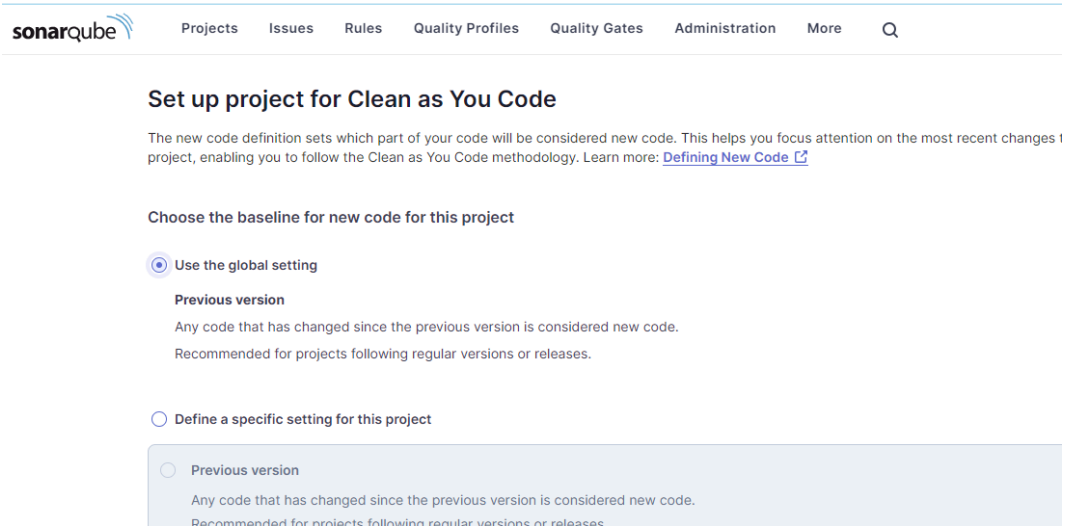


Next, we are supposed to Create a project. Give some suitable name to your project. A project will be generated for it automatically. We can change it or keep it the same

Name: Mohit Kerkar Div: D15C Roll No: 23



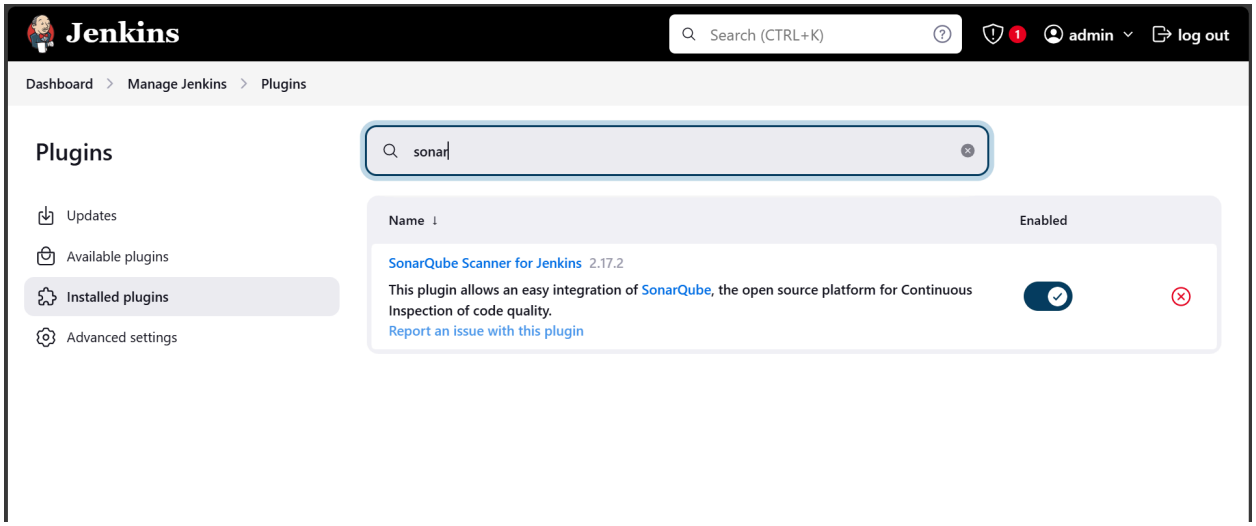
Choose global settings option for setting up your project



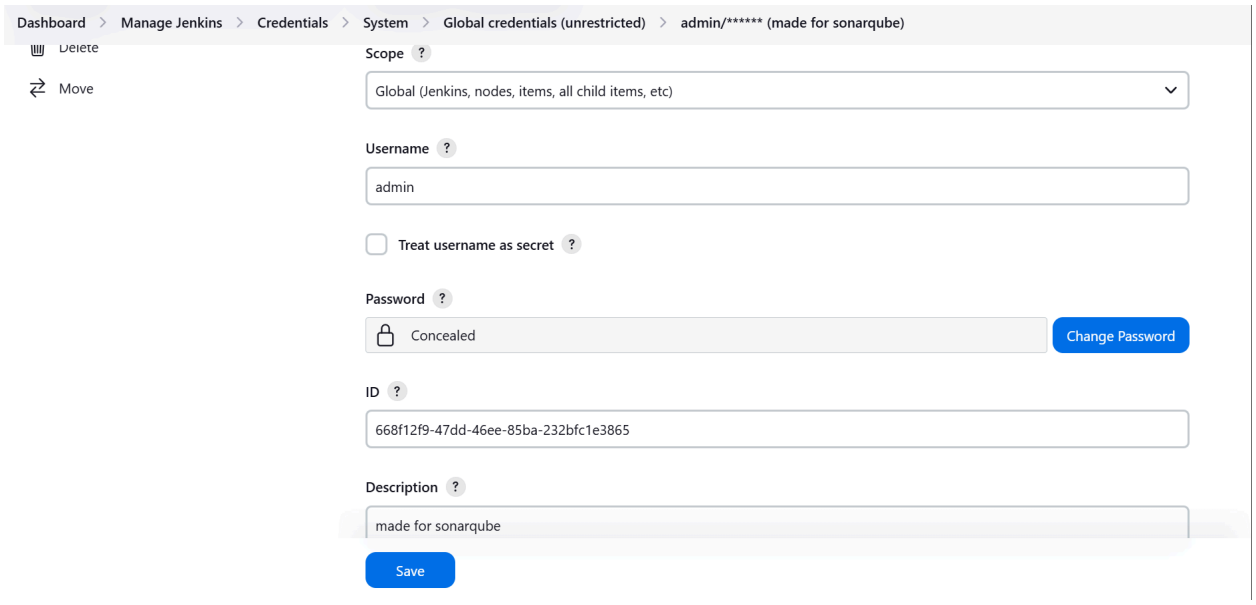
Now, on jenkins ..
For a few contents inside the script to be built using our pipeline, we need to configure those changes inside the global configuration settings inside Manage Jenkins.

First, Go to plugins inside Manage Jenkins and install the Sonarqube Scanner plugin which is necessary for the build to be executed.

Name: Mohit Kerkar Div: D15C Roll No: 23



Next, we will have to add our Sonarqube project specific credentials in the Credentials section of manage jenkins

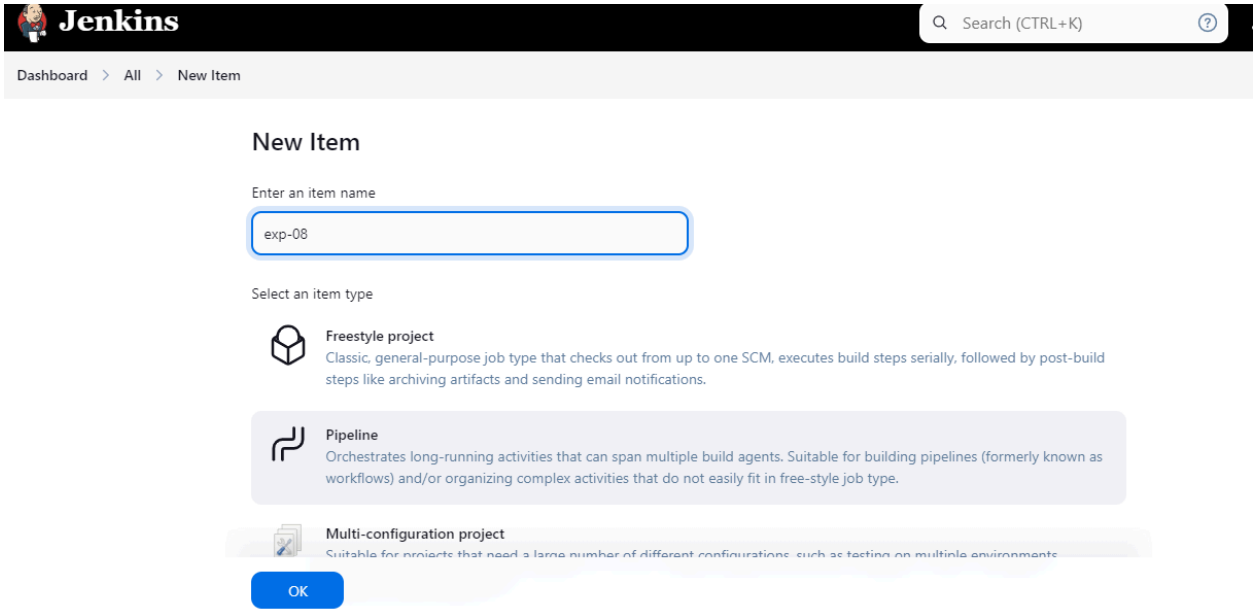


Here, i added the username i.e admin and my password … for which jenkins generated a automated ID for the credential block.

Now, our pipeline script will contain a line where we are supposed to provide the build with our source directory of the sonar-scanner batch file which we installed in the initial steps of our experiment
C:\sonarqube\sonar-scanner-6.2.0.4584-windows-x64\bin\sonar-scanner.bat
We will add this path inside the script to be built

Now, come to the dashboard
Create new item and select pipeline project. Give it a suitable name

Name: Mohit Kerkar Div: D15C Roll No: 23

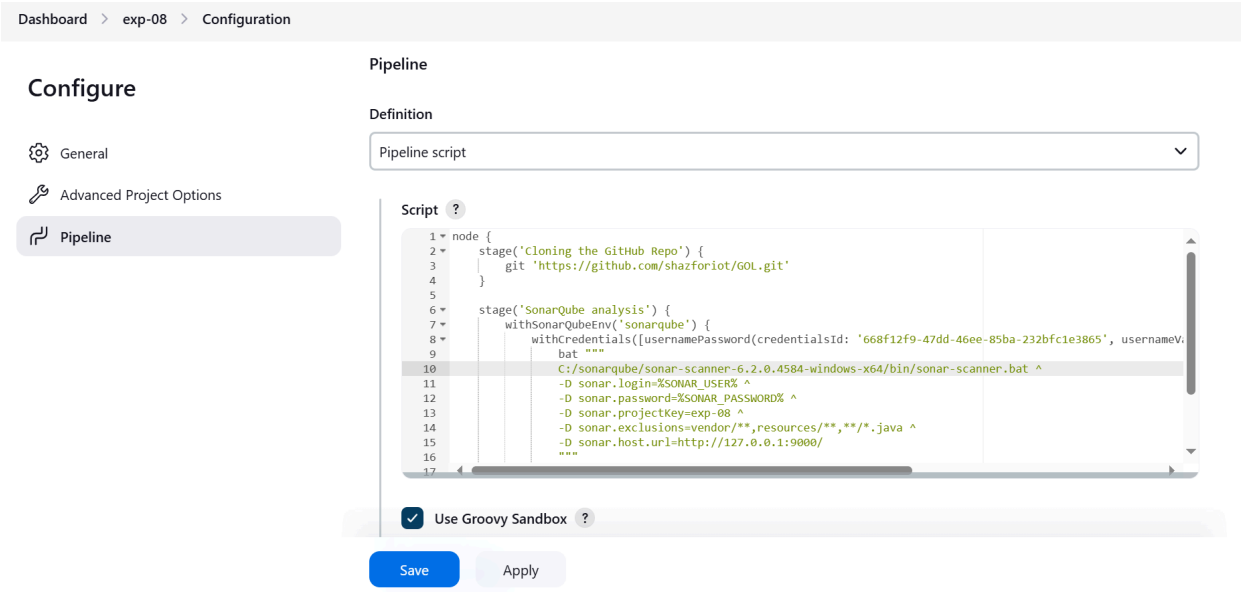


Put this script in the Pipeline Definition section and select the mode as Pipeline script

```
node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
   }

   stage('SonarQube analysis') {
      withSonarQubeEnv('sonarqube') {
         withCredentials([usernamePassword(credentialsId:
'668f12f9-47dd-46ee-85ba-232bfc1e3865', usernameVariable: 'SONAR_USER',
passwordVariable: 'SONAR_PASSWORD')]) {
            bat """
            C:/sonarqube/sonar-scanner-6.2.0.4584-windows-x64/bin/sonar-scanner.bat ^
            -D sonar.login=%SONAR_USER% ^
            -D sonar.password=%SONAR_PASSWORD% ^
            -D sonar.projectKey=exp-08 ^
            -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
            -D sonar.host.url=http://127.0.0.1:9000/
            """
         }
      }
   }
}
```

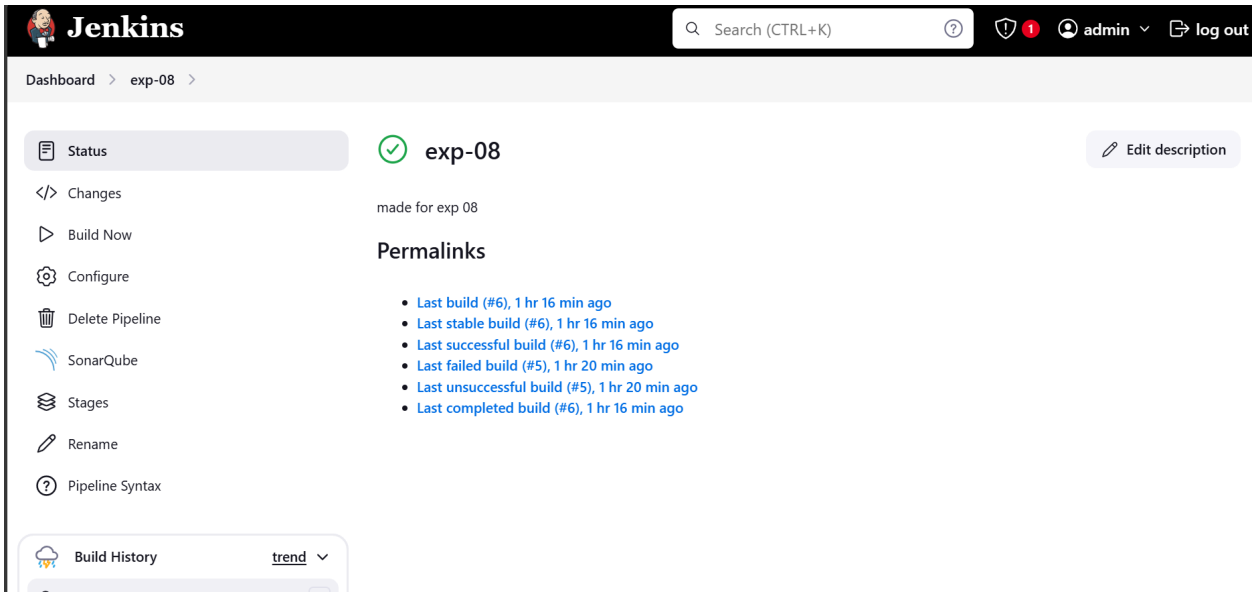
Name: Mohit Kerkar Div: D15C Roll No: 23



If you would notice carefully, i have used the withCredentials function within the pipeline script which makes the build avail the facility of using the credentials that i had set earlier

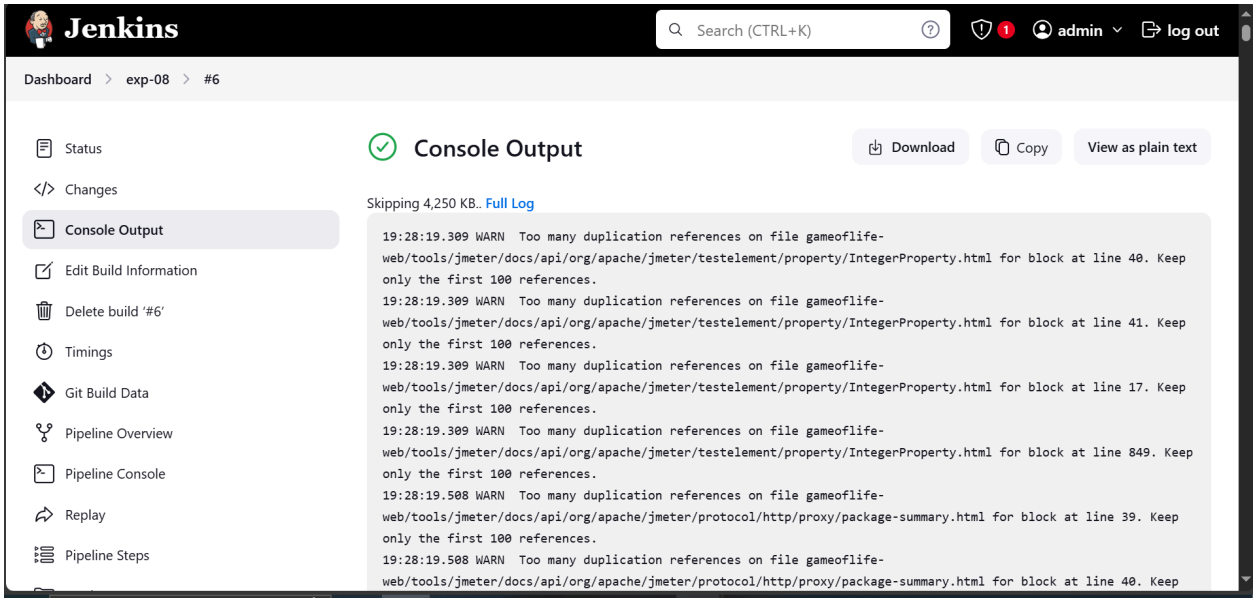
I have also added a line where i have mentioned the path of the sonar-scanner.bat file, since there are problems with system configurations with the installation of soanrqube

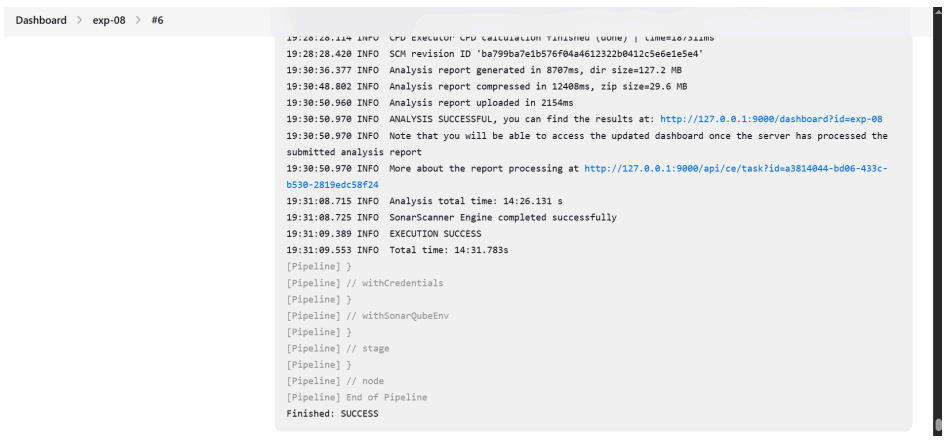
Now, save and build the pipeline



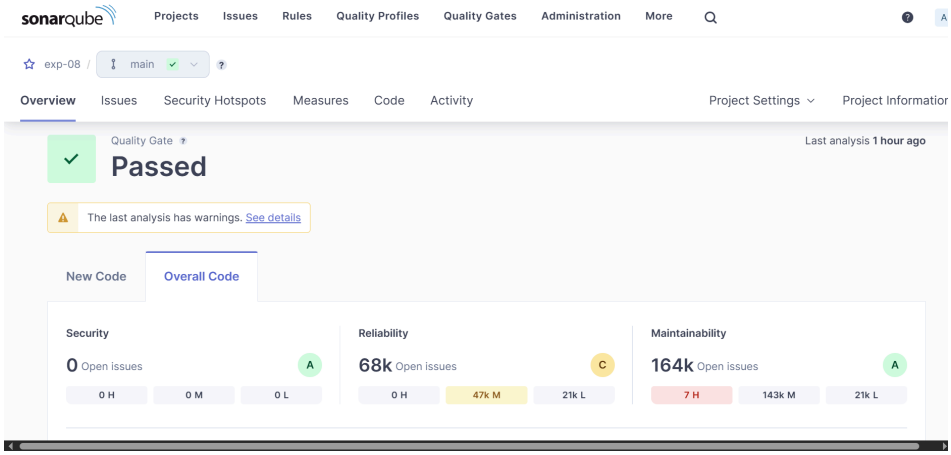
It generally takes 10-15 minutes for building this pipeline. We are supposed to stay patient and let the build process get completed

Name: Mohit Kerkar Div: D15C Roll No: 23





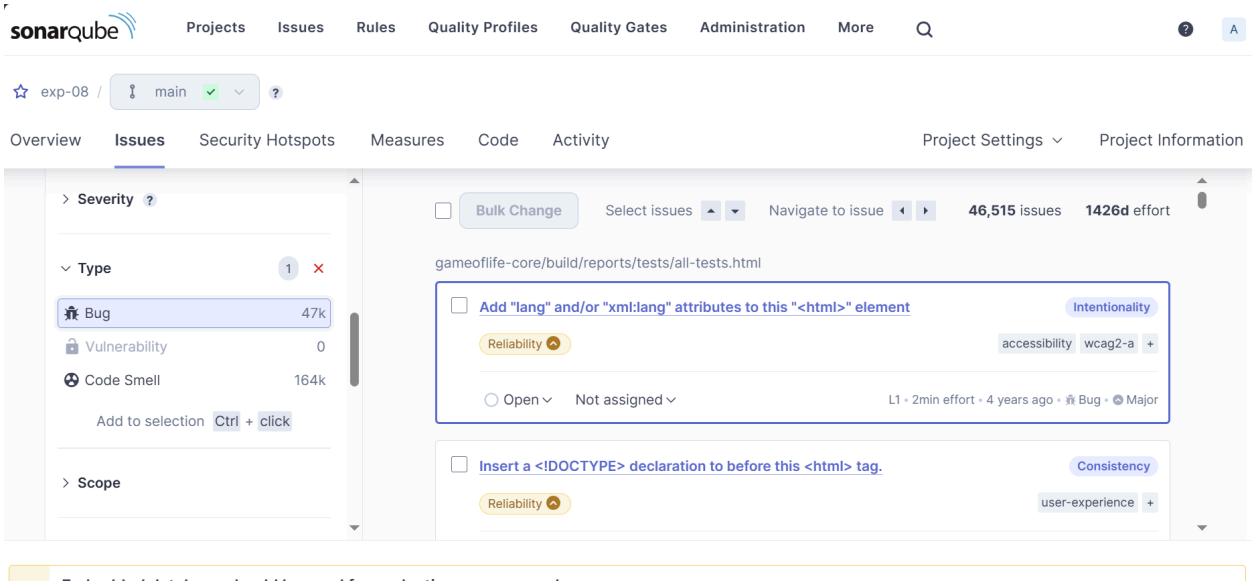
After a successful build, we are supposed to go back to our Sonarqube profile. We will see this type of a message on the screen, where it offers us multiple options to analyze our code



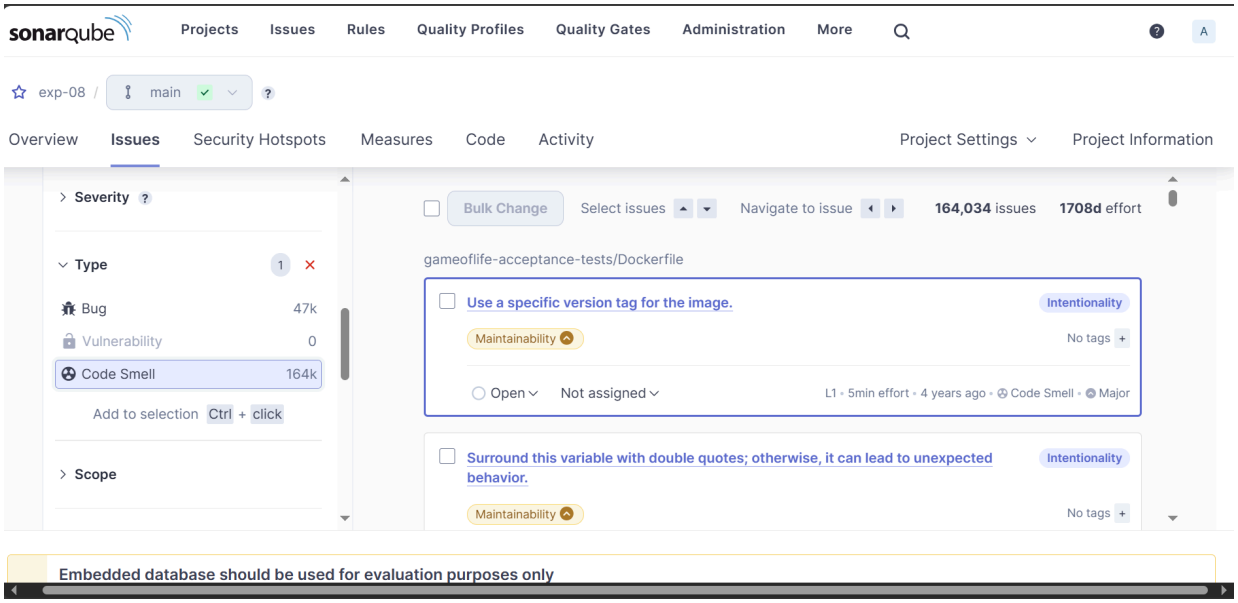
Within the issues section, let us explore different tabs

Name: Mohit Kerkar Div: D15C Roll No: 23

1. Bugs inside Type section

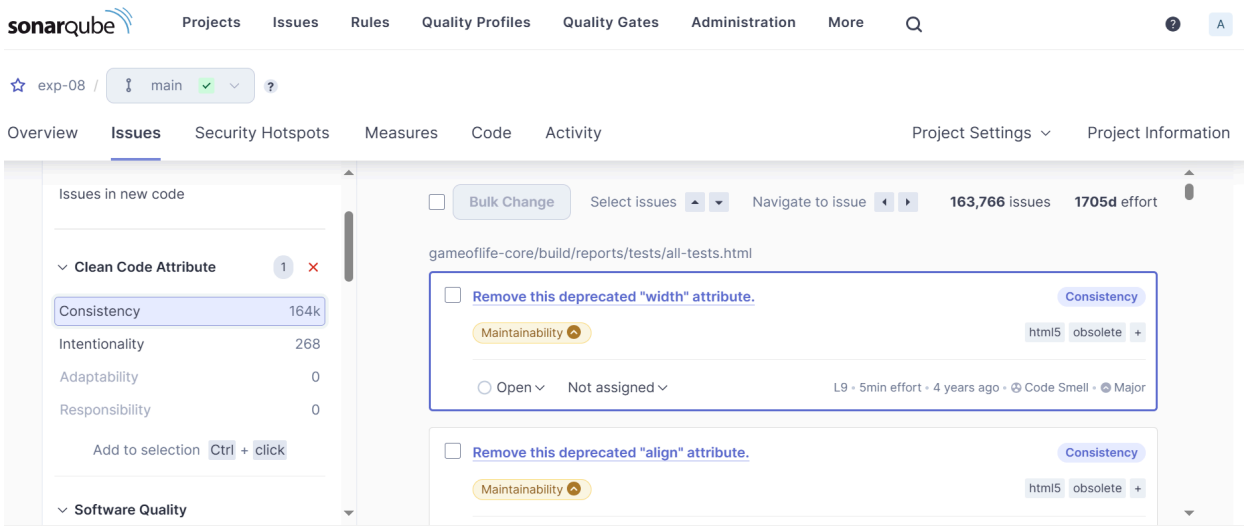


2. Code Smell inside the Type section

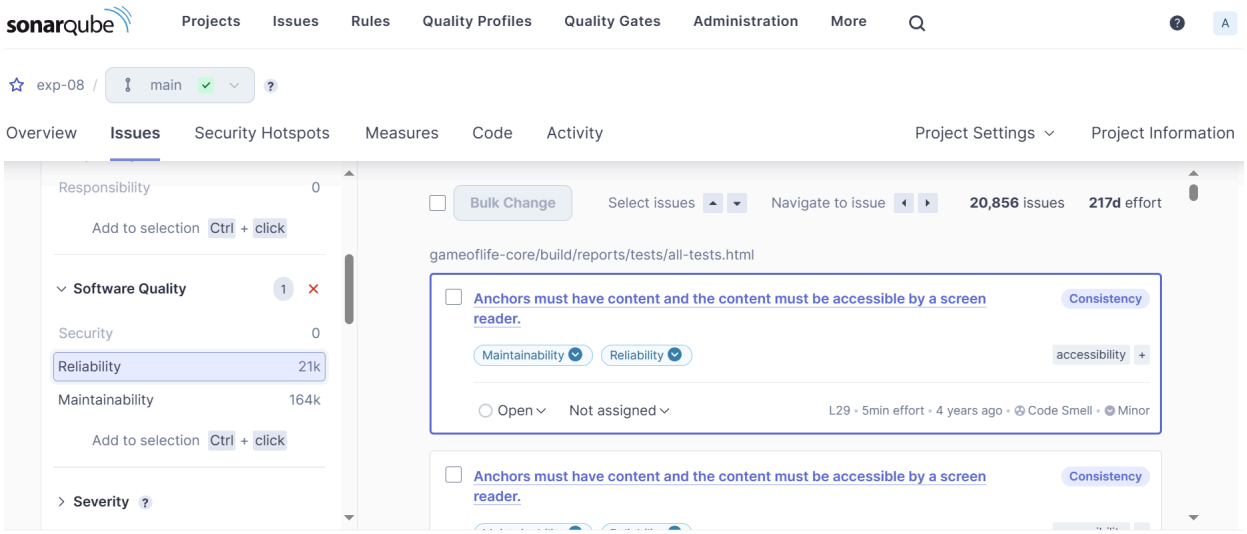


3. Consistency inside Clean Code Attribute

Name: Mohit Kerkar Div: D15C Roll No: 23

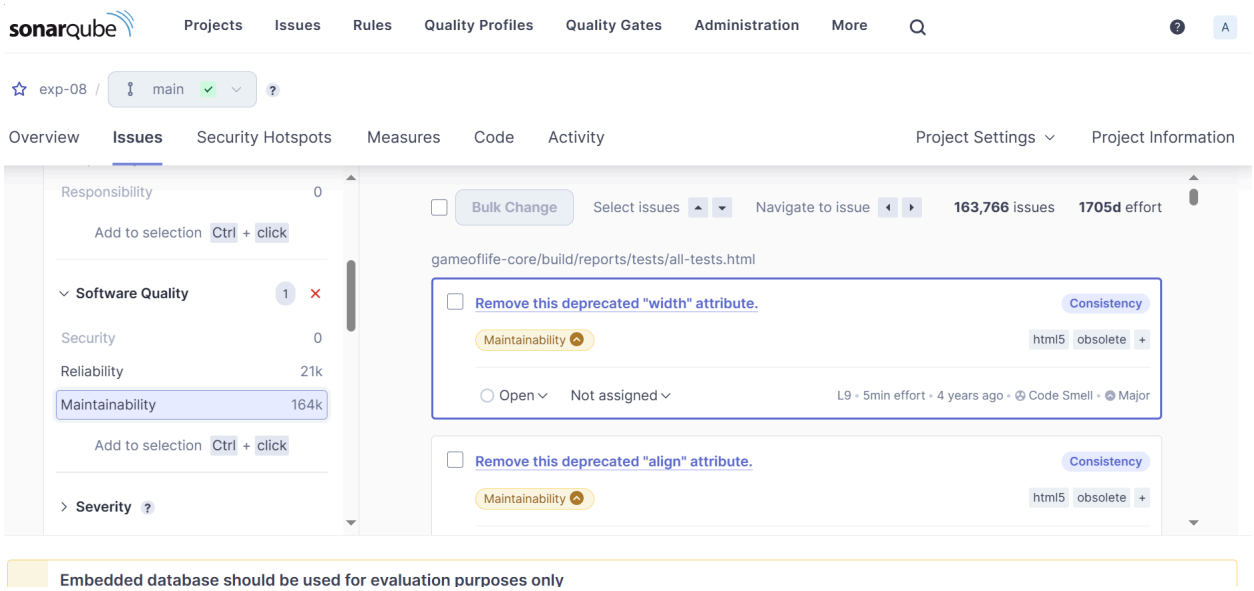


4. Reliability inside Software Quality



Name: Mohit Kerkar Div: D15C Roll No: 23

5. Maintainability



Conclusion:

In this experiment, we set up a Jenkins CI/CD pipeline integrated with SonarQube to automate static analysis on a sample application. Jenkins was configured to trigger builds and run SonarQube's analysis with every code change, detecting bugs, code smells, and security vulnerabilities. This pipeline provided continuous monitoring and ensured early detection of issues, improving code quality and security. The experiment showcased how integrating CI/CD pipelines with SonarQube enhances development efficiency and ensures better, more reliable software